CST1
COMPUTER SCIENCE TRIPOS Part IB

Monday 3 June 2024 13:30 to 16:30

COMPUTER SCIENCE Paper 4

*Answer **five** questions.*

*Submit the answers in five **separate** bundles, each with its own cover sheet. On each cover sheet, write the numbers of **all** attempted questions, and circle the number of the question attached.*

You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

STATIONERY REQUIREMENTS
Script paper
Blue cover sheets
Tags

SPECIAL REQUIREMENTS
Approved calculator permitted

1 Compiler Construction

Here are two grammars for languages built from terminals `a` and `b`:

Grammar A:

S	→	C a C b A
A	→	B
A	→	S
B	→	ϵ
B	→	b
C	→	ϵ

Grammar B:

S	→	A b
A	→	ϵ
A	→	A a b

In both grammars `S` is the start symbol.

(*a*) The languages L(A) and L(B) for grammars A and B are not equal.

(*i*) Give an example of a string that is in L(A) but not in L(B).

(*ii*) Give an example of a string that is in L(B) but not in L(A).

[2 marks]

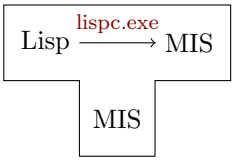
(*b*) Give regular expressions describing each of L(A) and L(B). [4 marks]

(*c*) Compute NULLABLE, FIRST sets and FOLLOW sets for each grammar. [6 marks]

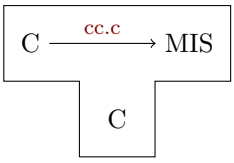
(*d*) State with justification whether each grammar is LL(1) and whether it is SLR(1). [8 marks]

2 Compiler Construction

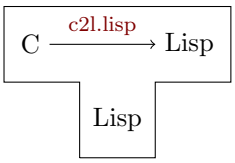
You have acquired an old computer, and you would like to write programs for it in C. Unfortunately, the computer has no executable C compiler: it has only a Lisp compiler `lispc.exe` (an executable that uses the machine's instruction set MIS):



and the source code for a C compiler `cc.c` (written in C):



(*a*) You would like to get the C compiler running efficiently on your computer with as little programming as possible. A useful first step is to write a quick-and-dirty C-to-Lisp compiler in Lisp:



What further steps are needed? Describe and illustrate each step using the same illustration scheme as above.

[8 marks]

(*b*) You would like to extend the C compiler to support foreign function calls so that your C programs can use libraries compiled by `lispc.exe`.

(*i*) What information from the *Application Binary Interface* might you need to ensure that foreign calls work correctly? Comment on function calls, on the stack, on object layout, and on any other relevant considerations.

[6 marks]

(*ii*) Programs written in Lisp use a garbage collector (GC) to reclaim unreachable objects. Explain how the GC determines reachability, how calls between C and Lisp might affect the reachability check, and what changes are needed to make the check work in the presence of those calls.

[6 marks]

3 Concepts in Programming Languages

(*a*) (*i*) Explain what a *monad* is in the context of a structure in a program. Define the two fundamental operations of a monad along with their types. You may use syntax from any programming language in your answer, but Haskell or OCaml are easiest. [2 marks]

(*ii*) Describe a particular monad and a program fragment that uses it when coding in Haskell. [4 marks]

(*iii*) Discuss briefly why monadic style programming may not be as necessary when coding in JavaScript or OCaml or Scala as it is in Haskell. [2 marks]

(*b*) For each of the following OCaml declarations, if they pass the type checker give their inferred types, or if not then give a program fragment using them that would violate type safety:

(*i*) `exception Exn of 'a` [2 marks]

(*ii*)
```
let rec even = function 0 -> true  | v -> odd (v-1)
    and odd  = function 0 -> false | v -> even (v-1)
```
[2 marks]

(*iii*) `fun a b -> a (a b)` [2 marks]

(*iv*) `fun a b -> b (a b)` [3 marks]

(*v*) `fun a b -> a b b` [3 marks]

4 Prolog

In your answers ensure each relation which you define has a comment giving a declarative reading of its behaviour. Avoid unnecessary use of *cut* or other extra-logical relations. The library relations `\=`, `is`, and `atomic(X)` may be used, the last succeeding if `X` is a number or atom e.g. `42` or `abc`. Other relations should not be assumed.

(*a*) Define a relation `eval/2` to reduce arithmetic terms, so that e.g. `eval(1+2*3,N)` succeeds with `N=7`. Atoms should reduce to themselves, so e.g. `eval(a,Ans)` succeeds with `Ans=a`. [4 marks]

(*b*) Extend `eval` to allow function calls within arithmetic terms, such that `2 * apply(inc, [2+3])` reduces to `12`. We will declare functions as Prolog facts in a `fun/2` relation, e.g. `inc` above will be specified by the fact `fun(apply(inc,[N]), N+1)`. Note the function arguments are held in a list, `[N]` in this example, to support multi-argument functions. [6 marks]

(*c*) Extend relation `eval` to support a `==/2` operator, which reduces term `A==B` to `true` if `A` and `B` reduce to the same number or atom and `false` otherwise. For example `eval(1+3==2+2,Ans)` succeeds with `Ans=true`. [3 marks]

(*d*) Extend relation `eval` to support terms of the form `if(Condition,Then,Else)`. These `if/3` terms should reduce to the reduction of either the `Then` term or the `Else` term determined by `Condition` reducing to `true` or `false`. For example `eval(if(1+2==4+5,a,b),Ans)` succeeds with `Ans=b`. Add a fact to the `fun` relation specifying the *factorial* function such that `eval(apply(fact,[5]),N)` succeeds with `N=120`. [7 marks]

5 Programming in C and C++

(*a*) Are C functions exactly the same as C++ functions? What is the difference between a C function and C++ method? [3 marks]

(*b*) The C language is used on a range of microcontrollers where the number of bits used for the `char` type might have any value between 8 and 16, but `int` always uses 24 bits. Write a program that determines the number of bits present in a `char`. [3 marks]

(*c*) On another computer, `char` variables are 8-bits in size, but it is not known if they are signed and unsigned. Write a program to test which. [3 marks]

(*d*) C++ is object oriented.

(*i*) Can C++ upcasts and downcasts always be checked for type safety at compile time? [3 marks]

(*ii*) Give an example of a C++ upcast that requires the object pointer to be adjusted. [3 marks]

(*iii*) Give an example where a C++ downcast is checked for validity at run time. What is needed for validity to be checked by the language runtime system? [3 marks]

(*e*) Say why and whether you would expect the following fragment to work:

```
char *mys = "ab_de";  mys[2] = 'c';
```

[2 marks]

6 Programming in C and C++

(*a*) A C programmer has an array of pointers to structs. They sort the array using the built-in comparison operator, '<'. Describe one circumstance where this will produce undefined behaviour and another situation where the behaviour is defined. [2 marks]

(*b*) Give three reasons why is it helpful to separate the declaration and implementation of classes in OO programming. The following code gives a (possibly incorrect) C++ signature declaration for a collection type. Criticise this definition and explain any difficulties that might arise with having the implementation in a separate file. If all types entered in the collection inherit from a common parent, can this make a difference? [8 marks]

```
template <typename T> class bucket
{ public:
  bucket(int N);     // Constructor: holds up to N items.
  int push(T *item); // Add item or return non zero if full.
  T pop();           // LIFO order pop most-recently added.
  T *dequeue();      // FIFO order dequeue of oldest item.
};
```

(*c*) Instead of holding pointers to objects, a collection class can hold the objects themselves. What changes in the `bucket` method types would be needed? What are the advantages and disadvantages of this change? [4 marks]

(*d*) Returning to holding references in the collection, it is instead required that all objects pushed must have a `void foo()` method that the `bucket` will invoke for all members currently in the `bucket` on either removal operation. Define how items might be stored inside the `bucket` and sketch suitable code for the `dequeue()` method with this addition. Can the `foo()` method be invoked using static or dynamic method invocation? [6 marks]

7 Cybersecurity

Some naïve teenagers want to attack a web application that does not salt its users' passwords, but simply hashes them. Hoping they will someday obtain the file of digests on the dark web, they want to precompute a compressed lookup table that will let them quickly crack all lowercase alphanumeric passwords of up to 15 characters once they get the file. They implement precomputed hash chains, but they ignore the possibility of collisions because they do not understand the issue. Their table must fit in their available disk space s. Each digest is 16 bytes long.

Please use the following symbol names and use sensible approximations where needed.

l_a	length of alphabet (charset size) for targeted passwords	$26 + 10 = 36$
l_d	length of one digest, in bytes	16
l_p	length of password, in bytes (max len to be explored)	15
s	storage space available for the compressed table, in bytes	8×10^{12}
t_h	time to compute a hash on attackers' CPU, in seconds	10^{-5}

(a) Under the attackers' incorrect assumption that hash collisions may be ignored:

(i) Derive formulae for the maximum number n_c of chains in the table, and for the minimum length l_c (in passwords) of each chain. Also calculate numerical values for these quantities. *(4 results.)* [4 marks]

(ii) Give a clear and full description of the algorithm for recovering the password that is the preimage of a given digest d. Pseudocode is not required for full marks, but will be rewarded if it adds clarity and precision. [5 marks]

(iii) Derive a formula for the minimum number n_h of hashes to be computed to build the compressed table, and calculate its numeric value. *(2 results.)* [2 marks]

(iv) Use the n_h value from Part (a)(iii) to calculate the time t_s needed to compute the whole table sequentially, assuming one hash computation takes time t_h on the attackers' CPU. Then imagine parallelising the work in three ways: moving from CPU to GPU, using a bank of GPUs, or using a distributed pool of collaborators. Give a reasonable speedup factor for each. Compute how long it would take to build the compressed table if all three speedups were adopted. *(5 results.)* [5 marks]

(b) Briefly explain what hash collisions are and why they matter. How would the presence of collisions invalidate the answer to Part (a)(iii)? [4 marks]

8 Cybersecurity

Consider the following hacking game. You are given access, as user `player`, to a 32-bit x86 machine. The C program `challenge` (setuid `boss`) accepts user input and contains a buffer overflow vulnerability. The stack is executable, there are no stack canaries in the binary and ASLR is turned off. When the `challenge` program is running, the first byte of its vulnerable buffer (always word-aligned) may appear in memory anywhere within a 32768 byte region starting at `0xffff6000`. The vulnerable buffer is 100 bytes long and the C function that contains it has only two more local variables, both `char`. The `challenge` program truncates its input at 2048 bytes.

Your input to the `challenge` program, including a specific 50-byte payload, is built by the following Python function. You supply five parameters to `buildInput` to create an input and then the `challenge` program is run on the resulting input, but doing so costs you \$0.01 per byte of generated input plus \$50 per run. You may repeat this as needed. You win a large prize if you cause the payload to be executed under userid `boss`. You seek a winning strategy that maximises your profit.

```
def buildInput(length, payloadOffset, retOffset, retAddress, retCopies):
    payload = bytes(b"\x31...") # 50 bytes of evil x86 machine code
    input = bytearray([0x90] * length) # prefill with nop
    input[payloadOffset:payloadOffset+len(payload)] = payload
    for j in range(retCopies): # stack spraying
        input[retOffset+(j*4):retOffset+(j*4+1)] = \
            retAddress.to_bytes(4, byteorder='little')
    return input
```

(*a*) Explain whether you would favour more runs or longer inputs. [1 mark]

(*b*) Explain whether, in your input, you would put the nop sled and payload before or after the stack spraying region. [3 marks]

(*c*) Explain whether you would favour a longer nop sled or a longer stack spraying region. [3 marks]

(*d*) Outline, with justification, an efficient strategy for guaranteeing a win in this game, detailing among other things how many *runs* (invocations of the `challenge` program) it involves in the worst case. [5 marks]

(*e*) Give, with justification, the specific values of each of the five input parameters your strategy will pass to `buildInput()` during run i, with runs numbered from 0 onwards. The values may be constant or parametric in i. [8 marks]

END OF PAPER